

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A modified transportable byte code execution environment comprising:
 - a substantially unmodified transportable byte code virtual machine;
 - a set of substantially unmodified base classes;
 - one or more overlays to the set of substantially unmodified base classes, the one or more overlays enabling corresponding base classes to support shared access by one or more substantially unmodified transportable byte code applications;
 - an unmodified primordial class loader for loading the system base classes as overlaid by the one or more overlays to the base classes;
 - a security manager supporting multiple applications and for limiting access to system resources according to user permissions, ~~wherein the multiple applications also have their own security management policies;~~ and
 - a dynamic class loader generator for creating a class loader for loading an application, the application classes and creating a thread group for the application;

wherein in response to an object of a first application requesting a resource of a shared base class, the one or more overlays are configured to identify properties of the first application by:

identifying a first class loader that loaded the object;

if the object was loaded by a first class loader that was created for the application,

using information in the first class loader and its associated namespace to

identify the properties of the first application; and

if the object was loaded by the primordial class loader, identifying a first thread group created for the first application and using information associated with the first thread group to identify the properties of the first application.

2. (Previously Presented) The modified transportable byte code execution environment of claim 1, wherein the application includes at least one of an application class loader and an application security manager.
3. (Previously Presented) The modified transportable byte code execution environment of claim 1, wherein the one or more overlays include overlays to file classes to limit access to system resources according to user permissions associated with the application.
4. (Cancelled)
5. (Previously Presented) The modified transportable byte code execution environment of claim 1, wherein the application includes one or more invocations of Abstract Window Toolkit (AWT) classes.
6. (Previously Presented) The modified transportable byte code execution environment of claim 1, wherein the one or more overlays support determining a calling application.
7. (Previously Presented) The modified transportable byte code execution environment of claim 6, wherein the determining a calling application comprises identifying the class loader of a calling method, and using the class loader to identify the application.
8. (Previously Presented) The modified transportable byte code execution environment of claim 6, wherein the determining a calling application comprises identifying the thread group for a calling method, and using the thread group to identify the application.

9. (Currently Amended) A method of supporting a number of applications in a single transportable byte code execution environment, the method comprising:

~~means for~~ generating a class loader for each of the applications in the number of applications, the class loader providing a name space for each application, and a thread group for each application;

~~means for~~ overlaying one or more substantially unmodified base classes to support the number of applications;

~~means for~~ determining a calling application for a method; and

~~means for~~ limiting access by the number of applications to system resources according to user permissions; and ~~, wherein the number of applications also have their own security management policies.~~

in response to an object of a first application requesting a resource of a shared base class:

identifying a first class loader that loaded the object;

if the object was loaded by a first class loader that was created for the application,

using information in the first class loader and its associated namespace to

identify properties of the first application; and

if the object was loaded by the primordial class loader, identifying a first thread

group created for the first application and using information associated

with the first thread group to identify the properties of the first application.

10. (Original) The method of claim 9, wherein at least one of the number of applications includes an application class loader.

11. (Original) The method of claim 9, wherein at least one of the number of applications includes an application security manager.

12. (Currently Amended) A computer data signal embodied in a carrier wave comprising:

a computer program for supporting a number of substantially unmodified transportable byte code applications on a substantially unmodified transportable byte code virtual machine, the transportable byte code virtual machine including a set of substantially unmodified base classes and a substantially unmodified primordial class loader, the program comprising:

a first set of instructions for generating a class loader for each of the transportable byte code applications in the number of substantially unmodified transportable byte code applications, the class loader providing a name space for each application, and a thread group for each application, the first set of instructions further associating a user with each application;

a second set of instructions for overlaying one or more substantially unmodified base classes to support the number of applications;

a third set of instructions for determining a calling application for a method; and

a fourth set of instructions for limiting access to a system resource by an application according to whether the user associated with the application has access to the system resource, wherein the application also has its own security management policies; and -

a fifth set of instructions for, in response to an object of a first application requesting a resource of a shared base class:

identifying a first class loader that loaded the object;

if the object was loaded by a first class loader that was created for the

application, using information in the first class loader and its

associated namespace to identify properties of the first application;

and

if the object was loaded by the primordial class loader, identifying a first

thread group created for the first application and using information

Appl. No. 09/464,352
Amdt. dated December 20, 2004
Reply to Office Action of June 21, 2004

associated with the first thread group to identify the properties of
the first application.

13. (Cancelled)